Amendments to the Claims:

1. (Currently Amended) A method for supporting a platform independent object format for a run-time environment, comprising the computer-implemented steps of:

accessing a definition of an object in terms of a composition of one or more primitive types;

accessing a platform-specific description of size and alignment of the one or more primitive types; and

generating a layout for the object in [['']] a high-order language based on the definition of the object and the size and alignment of the one or more primitive types.

2. (Original) The method according to claim 1, further comprising the step of generating instructions for an accessor operation to access a slot in the object holding a value for one of the one or more primitive types.

3. (Original) The method according to claim 1, further comprising the step of generating instructions for a get operation to fetch a value for one of the one or more primitive types from a slot in the object.

4. (Original) The method according to claim 1, further comprising the step of generating instructions for a set operation to store a value for one of the one or more primitive types from a slot in the object.

5. (Previously Presented) The method according to claim 1, wherein the one or more primitive types includes one or more of the following types: integer, floating point, and reference.

6. (Original) The method according to claim 5, wherein the primitive reference type is one of a native machine pointer type and a numeric reference type.

7. (Canceled)

8. (Original) A method for supporting an object format for a plurality of incompatible platforms for a run-time environment, comprising the computer-implemented steps of:

accessing a definition of an object as a plurality of slots containing a primitive type;

accessing a plurality of platform-specific descriptions of layout parameters of the one or more primitive types, said platform-specific descriptions corresponding respectively to the incompatible platforms; and

generating a plurality of layouts, corresponding respectively to the incompatible platforms, for the object in a high-order language based on the definition of the object and the platform-specific descriptions.

9. (Original) The method according to claim 8, where the slots are located in the layouts for the incompatible platforms, when compiled by a corresponding platform-specific compiler, at same offsets.

10. (Previously Presented) A computer-readable medium bearing instructions for supporting a platform independent object format for a run-time environment, said instructions being arranged to cause one or more processors upon execution thereby to perform the steps of:

accessing a definition of an object in terms of a composition of one or more primitive

types;

accessing a platform-specific description of size and alignment of the one or more primitive types; and

generating a layout for the object in a high-order language based on the definition of the object and the size and alignment of the one or more primitive types.

11. (Original) The computer-readable medium according to claim 10, wherein said instructions are further arranged for performing the step of generating instructions for an accessor operation to access a slot in the object holding a value for one of the one or more primitive types.

12. (Original) The computer-readable medium according to claim 10, wherein said instructions are further arranged for performing the step of generating instructions for a get operation to fetch a value for one of the one or more primitive types from a slot in the object.

13. (Original) The computer-readable medium according to claim 10, wherein said instructions are further arranged for performing the step of generating instructions for a set operation to store a value for one of the one or more primitive types from a slot in the object.

14. (Previously Presented) The computer-readable medium according to claim 10, wherein the one or more primitive types includes one or more of the following types: integer, floating point, and reference.

15. (Original) The computer-readable medium according to claim 14, wherein the primitive reference type is one of a native machine pointer type and a numeric reference type.

16. (Canceled)

17. (Original) A computer-readable medium bearing instructions for supporting. an object format for a plurality of incompatible platforms for a run-time environment, said instructions being arranged to cause one or more processors upon execution thereby to perform the steps of:

accessing a definition of an object as a plurality of slots containing a primitive type;

accessing a plurality of platform-specific descriptions of layout parameters of the one or more primitive types, said platform-specific descriptions corresponding respectively to the incompatible platforms; and

generating a plurality of layouts, corresponding respectively to the incompatible platforms, for the object in a high-order language based on the definition of the object and the platform-specific descriptions.

18. (Original) The computer-readable medium according to claim 17, wherein the slots are located in the layouts for the incompatible platforms, when compiled by a corresponding platform-specific compiler, at same offsets.

19. (Previously Presented) The method according to claim 9, wherein:

one of the platform-specific descriptions corresponding to one of the incompatible platforms specifies that the primitive type has a first size;

another of the platform-specific descriptions corresponding to another of the incompatible platforms specifies that the primitive type has a second size greater than the first size; and said generating the layouts includes:

generating one of the layouts corresponding to said one of the incompatible platforms; generating another of the layouts corresponding to said another of the incompatible platforms; and

generating a padding element in said one of the layouts.

20. (Previously Presented) The computer-readable medium according to claim 18, wherein:

one of the platform-specific descriptions corresponding to one of the incompatible platforms specifies that the primitive type has a first size; and

another of the platform-specific descriptions corresponding to another of the incompatible platforms specifies that the primitive type has a second size greater than the first size; and said generating the layouts includes:

generating one of the layouts corresponding to said one of the incompatible platforms;

generating another of the layouts corresponding to said another of the incompatible platforms; and

generating a padding element in said one of the layouts.

21. (Previously Presented) A method for supporting an object format for a run-time environment, comprising the computer-implemented steps of:

accessing a definition of an object as including at least one slot containing a primitive type;

accessing a first layout description for the primitive type corresponding to a first platform;

generating a first layout for the slot of the object in a high-order language based on the definition of the object and the first layout description; and

accessing a second layout description for the primitive type corresponding to a second platform; and

generating a second layout for the slot of the object in a high-order language based on the definition of the object and the first layout description;

wherein the first layout for the slot and the second layout for the slot have a same size when compiled by a first compiler of the high-order language on the first platform and a second compiler of the high-order language on the second platform; and

the first layout for the slot of the object in the high-order language includes a padding element and the second layout for the slot of the object in the high-order language does not include me padding element.

22. (Previously Presented) The method according to claim 21, wherein:

the first layout description for the primitive type specifies a first size for the primitive type; and

the second layout description for the primitive specifies a second size for the primitive type greater than the first size.

23. (Previously Presented) The method according to claim 21, wherein:

the first layout description for the primitive type specifies a first alignment restriction for the primitive type; and

the second layout description for the primitive specifies a second alignment restriction for the primitive type stricter than the first alignment restriction.